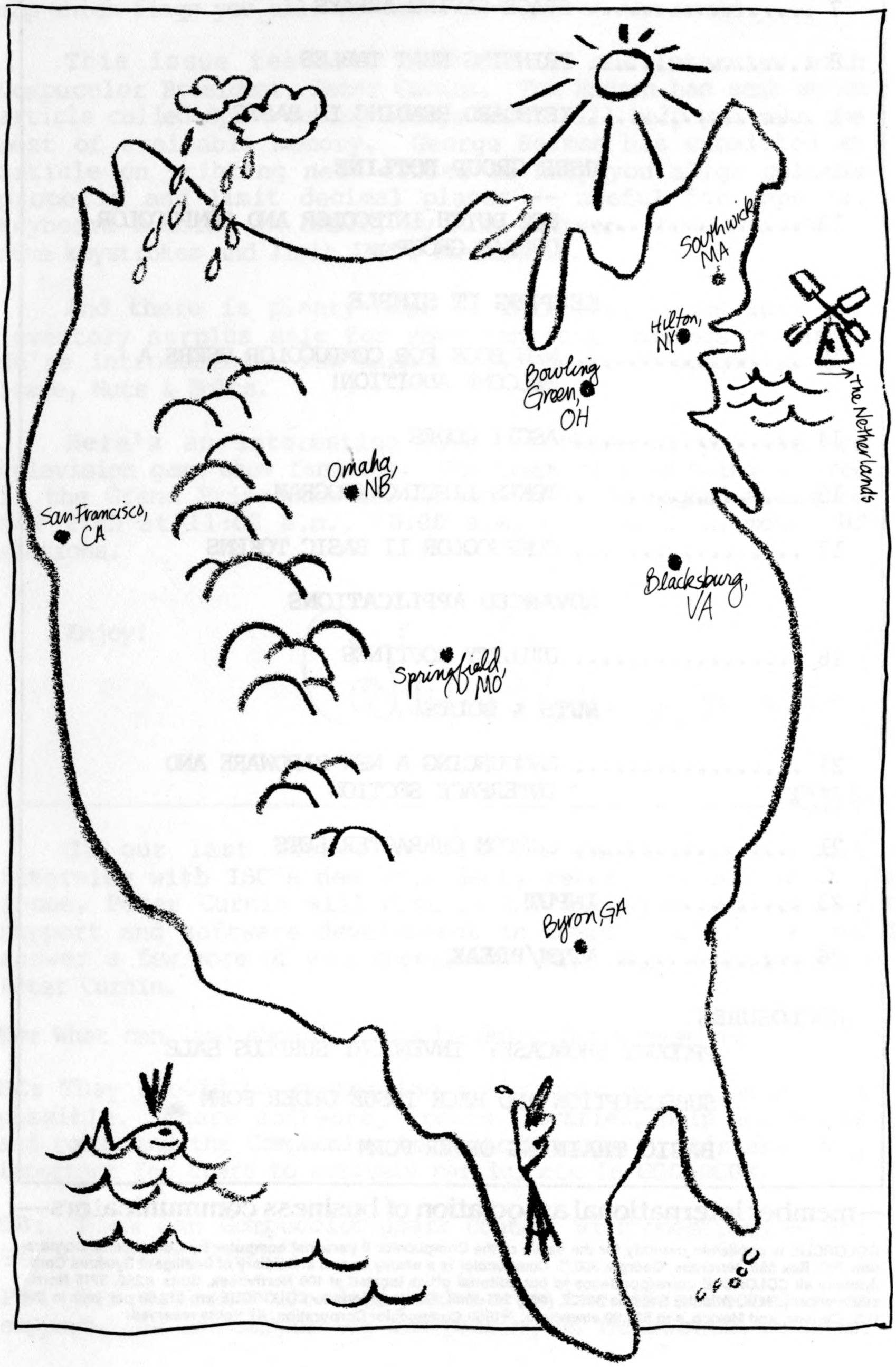
colorcue

a publication for Compucolor users · v. 3, · # 4 · april 1980 $1



inside

colorcue

menu

contributing to the success of this issue

USER INPUT
Tom Hudson
George Herman
Alan Matzger
Alberto Nogueras
Joseph Charles
Dennis Martin
R.M. Manazir
Trevor Taylor
Bill Greene

TECHNICAL ADVICE
Don Baines
Knox Pannill
Bruce Williams

ART DIRECTOR
Debbie Feldman

COVER
Jennifer Abramson

EDITOR
Cathy Abramson

—member international association of business communicators—

COLORCUE is published monthly for the users of the Compucolor II personal computer by Compucolor Corporation, P.O. Box 569, Norcross, Georgia 30071. Compucolor is a wholly owned subsidiary of Intelligent Systems Corp. Address all COLORCUE correspondence to our editorial office located at 100 Northcreek, Suite #250, 3715 Northside Parkway, N.W., Atlanta, Georgia 30327, (404) 261-3003. Subscriptions to COLORCUE are $12.00 per year in the U.S., Canada, and Mexico, and $24.00 elsewhere. ©1980 Compucolor Corporation. All rights reserved.

editor's letter

Compucolor salutes you, our many users. You come from every corner of the globe -- from as close to our plant as Atlanta and from as far away as Australia. This issue has been almost entirely written by you. Each contributor to this issue has been flagged on our cover map. If we were to show you a map which flags you all, it would be a map of the world.

This issue features Part Two of our interview with Compucolor President, Peter Curnin. Tom Hudson has sent us an article called Space-Saving Arrays which will help you make the most of available memory. George Herman has submitted an article on printing neat tables to help you align columns properly and limit decimal places -- useful for reports. Keyboard Reading in BASIC, by Alan Matzger, shows you how to save keystrokes and limit INPUT statements.

And there is plenty more of interest. Look into our inventory surplus sale for some fantastic savings on parts! We're introducing a new hardware and interface column in this issue, Nuts & Bolts.

Here's an interesting note for those of you who are television game show fanatics. The Compucolor is being offered in the Grand Prize Showcase on the "Price Is Right" airing on May 27th at 11:00 a.m., 10:00 a.m. Central, on most CBS stations.

Enjoy!

Cathy Abramson

rem

In our last issue, we ran the first of a two-part interview with ISC's new President, Peter Curnin. In this issue, Peter Curnin will discuss his views on the role of support and software development in Compucolor's future and answer a few more of your questions. Part Two: Interview with Peter Curnin.

INTERVIEW WITH PETER CURNIN: PART TWO

ED: What can, and should, users be doing for themselves?

PC: They should be supporting their user groups as much as possible. Share software, create libraries, help new owners and recommend the Compucolor to their friends. It's also very important for users to actively participate in **COLORCUE.**

ED: Whom can Compucolor users contact with questions -- and get a straight answer?

PC: Our dealers are designed to be the first echelon of support. We are supporting and passing on information to them.

COLORCUE is another form of our support after the sale. And, to the degree that we can put in tips, address common problems that people have, and pass on solutions to them, **COLORCUE's** the most effective vehicle we have right now.

Share software, create libraries, help new owners and recommend Compucolor to your friends.

ED: How about the role of software in the future? Our users are very concerned about software development.

PC: The computer itself is limited by the number of applications and the uses that it's put to. I think to encourage people to develop programs, and let them get a financial reward out of it, is a very important thing to do. A number of these freebie 9 line or 20 line programs currently held in the User Software File may not be accomplishing what a lot of people want. I'm going to do all that I can to encourage enterprising programmers to sit down and do a nice piece of work with the idea that there is a financial reward to be gained. I don't want to frustrate someone who wants to develop software and yet has no way to reach other people who own Compucolors. If he or she wants to develop it, I want to cooperate. I want to help that happen.

ED: Are you looking for outside authors to design and develop software specifically for the Compucolor?

PC: Yes. We are going to make a very deliberate effort to promote software for both Compucolor and Intecolor.

Incidentally, the better programs come from what I would call a "cottage trade industry". They can be a person in their home or a husband and wife team. The people that are installing systems, the successful firms, have one to perhaps five employees; I would say that the average size is three. Many are installing from twelve to twenty systems a year. A very respectable business is installing one every other month. These small software houses are very effective.

So I go to the little shop. I have as much security with them as going downtown with some major corporation, having the one guy that's any good quit, and finding that no one in the building knew what he was doing! To me, there's no extra security being associated with a fifty man software development company.

ED: What support do you envision for special interest groups, business and education, for example?

PC: We want to be as helpful as possible to special interest organizations. We need people to state what needs to be done. We will attempt to work with what I'll call vertical markets -- any place where there is an opportunity to go in and make our

product more useful to that particular market segment. Those folks know more about their field than we do, obviously. We know, for example, that education now is learning a lot about computers. If somebody is learning about computers, or teaching children something that they're going to use for 25-30 years, I am sure that they would prefer to use an advanced computer, rather than a game. We have a variety of languages that work on the Compucolor -- it's a very sophisticated machine. The fact that we make a graphics color computer, and the fact that IBM is just moving in on it shows that if you learn color graphics now, you're learning the most advanced concepts around. You're learning what IBM says the future is going to be, for less than $2,000. Perhaps a school can teach 100 young people each year for the next five years what they need to function effectively for the next thirty years.

...if you learn color graphics now, you're learning the most advanced concepts around.

ED: I was reading an article recently in one of the education journals, and they were talking about Computer Assisted Instruction. The article said that the problem was not that teachers were unwilling to use the computer but that they were not being taught in college how to program or being given any instruction on what a computer is and how to use it -- that our main chore right now is not getting them interested in buying the computer, but training them how to use this tool once they have it.

PC: I don't think that's the answer, quite frankly. The computer in the education area, should become invisible. The role of the teacher should not have to be to understand computers. I want us to be working with people who know curriculum, and know what is required. Then we can interject a group of professionals to make a package out of it. But it should arrive in the classroom as a friendly device. It has to be an unobtrusive acquaintanceship between the teacher, the student, and the computer.

To say that before you can do this, you must know BASIC or APL is where the industry was before, and where we failed. I think when it succeeds, the computer will arrive integrated into the system. It will arrive with the exercise book and the exercises will be in it. There is no more reason to ask the teacher to write the computer book than there is to ask her to write the math, science or any of the other books. There will be a group of professionals that know both teaching and something about computers, but that's only two percent. Ninety-eight percent of the educational community should receive it in finished form. The important question is whether the curriculum is being presented in the finest form.

The computer should arrive in the classroom as a friendly device.

ED: Do you think that's analogous to the business situation now? That businessmen should not have to learn programming to use the machine?

PC: Yes. It's enough for someone to even understand the accounting system! You know, years ago you could go ahead and have two or three $25,000 people who did no other work but set up programs, and that was $50,000 a year. Now we're getting down to a computer that sells for $5,000. If you go up to someone and tell them that they need two $25,000 people to set them up -- not to put the payroll data in or anything else -- just to interface to your machine, it doesn't make for inexpensive computers.

ED: Here are some more questions from our readers. Chris Carson of Aurora, Colorado asks: "Several people in my user group would like to use the Compucolor in a ham radio environment, but the Compucolor generates too much noise. The FCC is trying to implement new regulations which would force manufacturers to control the amount of interference electronic equipment generates. How does the company intend to overcome this problem?"

Businessmen shouldn't have to learn programming to use the computer...it's enough for someone to even understand the accounting system!

PC: The Compucolor is being tested right now in an RF and EMI lab. When the testing is complete and recommendations are made, we will know how to proceed.

ED: Peter Russell of St. Lambert, Quebec, Canada wants to know: "Are there any courses being offered by Compucolor on the use of peripherals and I/O interfacing?"

PC: We do not offer such a course at this time. We do publish interface information in **COLORCUE** and will be expanding coverage in this area in issues to come.

ED: Tom Schenck of Fresno, California asks: "I think that the Compucolor can run circles around any other machine on the market. However, there are no programmers in our user group so we must rely on preprogrammed disks. We read about the availability of software for other competitive machines. Is Compucolor's only source of software Compucolor? What are you going to do about software availability?"

PC: This newsletter and user groups, such as the Compucolor Education Users' Group, have loads of interesting programs that are available for a few dollars or free in many cases. There is a little reaching out required, but there is a lot around.

=

SPACE-SAVING ARRAYS

By Tom Hudson
2369 E. Seminole
Springfield, Mo.
65804

Sooner or later, every programmer runs into a very fundamental problem: running out of memory. It may be a lot of REMs, inefficient code, or just a large program. In my case, it is the use of many large arrays. The numeric variable storage method used by most microcomputer systems uses four bytes of memory per variable. In a 64 by 32 numeric array, the memory used is over 8000 bytes (64 * 32 * 4 = 8192). There are many ways to conserve memory. You can eliminate all your REMarks (which makes a complex program difficult to de-bug when it blows up); you can eliminate all the frills (which usually makes the program impossible for anyone but you to use); or you can just give up (which doesn't do anybody any good). There is one other alternative, which I will cover in this article.

Let's say, for example, that you are writing the ultimate space war game. In order to represent the Galaxy in computer memory, you require a 100 by 100 member numeric array. A 0 in any array location will indicate empty space. A 1 will indicate a star. A 2 will indicate your spaceship. Simple arithmetic tells you that this array will normally require 40,000 bytes (100 * 100 * 4). Since you only have a 16K machine, this array is clearly impossible to have, using the system array routines. Using the method I will outline in this article, this same array will only require 10,000 bytes (100 * 100), and will easily fit in a 16K machine. The only drawbacks with this routine are a slight decrease in speed, and the fact that only positive integers up to and including 255 can be stored in this manner.

Here's the routine:

```
1 REM  ******** SPACE-SAVING ARRAYS ********
2 REM
3 REM  BY TOM HUDSON
4 REM  2369 E. SEMINOLE
5 REM  SPRINGFIELD, MO 65804
6 REM  417-883-0727
7 REM
8 REM  ** THIS PROGRAM DEMONSTRATES A 50 BY 40 ARRAY **
9 REM  ** 'DIMENSION' THE ARRAY **
10 D1= 50:D2= 40
15 REM
20 REM ** MOVE END OF BASIC BACK TO ACCOMODATE ARRAY **
30 AD= PEEK(32941)* 256+ PEEK(32940)
40 AD= AD- (D1+ 1)* (D2+ 1)- 202
50 POKE 32941,INT (AD/ 256):POKE 32940,AD- INT (AD/ 256)*
   256
55 REM
60 REM  ** SET UP BASE ADDRESS POINTER **
70 BA= AD+ 1
75 REM
80 REM  ** SET UP DISPLACEMENT FUNCTION **
90 D3= D2+ 1:DEF FN D(X)= BA+ S1* D3+ S2
95 REM
100 REM  ** LOAD ARRAY **
110 FOR S1= 0TO D1:FOR S2= 0TO D2
120 POKE (FN D(X)),(S1+ S2)
```

```
125 PRINT FN D(X),S1+ S2:REM  ** PRINT ADDRESS,
    CONTENTS **{
130 NEXT S2,S1
140 REM
145 REM  ** PRINT ARRAY **
150 PRINT "*** ARRAY VALUES FOLLOW ***":FOR T= 1TO 1000:
    NEXT T
160 FOR S1= 0TO D1:FOR S2= 0TO D2
170 PRINT FN D(X),PEEK (FN D(X)):REM  ** PRINT ADDRESS,
    CONTENTS **
180 NEXT S2,S1
190 REM
200 REM  ** PUT END OF BASIC BACK WHERE IT WAS **
210 AD= AD+ (D1+ 1)* (D2+ 1)+ 202
220 POKE 32941,INT (AD/ 256):POKE 32940,AD- INT
    (AD/ 256)* 256
230 END
READY
```

How It Works

First, you set up the dimensions of the array (D1,D2) in line 10. The array is positioned in high memory above BASIC. It gets this space by altering the high memory pointers in memory locations 32940 and 32941. The new high memory location + 1 is the BASE of the array. All array locations are calculated from the base. In order to find a given location of an array member, a user-defined function, FND(X), is used. This function calculates the DISPLACEMENT of an array member in memory (BASE + DISPLACEMENT = LOCATION). By PEEKing and POKEing with the FND(X) parameter, the routine simulates the normal array function. S1 and S2 are the subscript variables. They are the same as X and Y in the statement: N = A(X,Y). To store an amount, set up S1 and S2 with the proper subscripts, and execute a POKE FND(X), N (where N is another variable or any amount you wanted to store). To retrieve it, simply set up S1 and S2 as before, and execute a PEEK(FND(X)).

Once again, this method can only store positive integers up to and including 255. Any attempt to exceed these values will result in a CF ERROR. When used properly, this routine will enable you to go far beyond the normal limitations of system array routines, and will add a whole new dimension to your Compucolor II.

=

PRINTING NEAT TABLES

By George Herman
Bowling Green
State University
Bowling Green,
Ohio 43402

BASIC calculates numbers very nicely, but it prints them out in a fashion totally unsuited for neat tables and reports. For example, the little program in Listing 1 generates positive and negative numbers of several absolute sizes. Run on a Compucolor II, it prints out something like Table I on the screen, when what we were hoping for was the neat appearance of Table II.

By using the string functions built into CCII BASIC, the output of Table II can be produced. Listing 2 rewrites the little program with a subroutine to control the number of decimal places printed (and supply trailing zeroes where

necessary) and a PRINT statement which right-justifies the column of numbers. If you want to peek now, go ahead, but the next few paragraphs explain how it all happens.

First of all, line 120 sets D equal to the number of digits we want to print after the decimal point. It might seem that we could obtain that by a statement like PRINT INT(10^D*N)/10^D. If you try that, you will get no more than D decimal places, but some numbers will print out to only one or two places; some numbers won't even show the decimal point. What we do, then, is convert INT (10^D*N) to a string using the STR$(X) function. Now we can break it into substrings and put the decimal point in D digits from the right end of the string.

Before we do that, however, we have to take care of the leading zeroes of numbers smaller than 0.1, which vanished when we multiplied by 10^D. If we have used N$=STR$(INT(10^D*N+.5) to get a string, the first element of the string will be the minus sign for negative numbers and a blank space for positive ones. To get the zeroes in after the sign and ahead of the digits, we cut the string in two. The sign (or space) is saved in SN$=LEFT$(N$,1) and the digits in NN$=RIGHT$(NN$,LEN(NN$)-1). Now we can add zeroes in front of the digits until the string is as long as D digits. That's what line 10050 of Listing 2. is about. Now we are ready to put in the decimal point, so we will cut NN$ into D$ which has D digits, and I$ which has the integer part of N represented as a string. Of course, if N was less than one there is nothing in I$. That's why there is an IF in line 10070.

If all has gone well, we now have SN$ with a minus sign or space, I$ with the digits left of the decimal place and D$ with the digits to the right of the point. Line 10080 puts them all together in a new NN$ which is returned from the subroutine. Since every NN$ produced will have the same number of decimal places, line 160 will line up the decimal points neatly, one over the other as it prints down the column.

Listing 2 and Table II show the result of choosing two decimal places. By changing line 120, one can have more or fewer. However, if you set D equal to zero, the program will crash at line 10060 because RIGHT$ function will not accept a zero argument. Furthermore, if D is too big (more than 4 for this particular set of numbers) then line 10020 will yield some numbers in scientific (exponent) notation, which makes a mess of the output! Numbers too big to be handled without the exponent notation call for a different approach.

The subroutine in Listing 2 can be tacked on to other programs; pass numbers to it in N and get them back for printing as NN$, but don't forget to set D to a small positive number somewhere before the first GOSUB.

Incidentally, since you have the sign split off in SN$, you can alter the format of negative numbers to suit the bookkeeping style you prefer. For business reports, you may want to use

```
10080 NN$= I$+"."+D$+"  "+SN$
```

which puts the minus sign in a neat column to the right of the numbers. Or substitute

```
10080 P$=" ":PP$=" "
10082 IF SN$="-" THEN P$="(":PP$=")"
10084 NN$= P$+I$+"."+D$+PP$
```

and the negative numbers will be printed in parentheses. The spaces have to be tacked on to positive numbers to keep the columns justified. A similar trick using color control codes in P$ and PP$ in line 10082 permits printing the negative numbers in red and positive in green. If you try that, remember that the non-printing color codes do add one to the length of the string, just like the parentheses!

You have probably realized that the addition of 0.5 before taking the INT function is a trick to round to the nearest digit, instead of just dropping the last few digits. You may not wish to do this, in which case you can just omit the +0.5 from line 10020.

Even when you are not printing tables or lining numbers up in columns, give some thought to trimming the number of digits after the decimal place. BASIC will give you all the digits it can wring out of a computation. Results may show up to five or six decimal places, when the data on which the calculations were performed were only accurate to two places. The spurious appearance of precision in approximate results is misleading.

Anyhow, there are some numbers that don't make sense to more than a few places. You probably don't want to be told that for your age and height your weight should be 147.89365 pounds. My wife doesn't want me to tell her that for a given yardage of finished fabric, she needs 253.84972 yards of yarn before she starts weaving. With the subroutine in Listing 2 numbers can be kept looking about as they are expected to look.

TABLE I	**TABLE II**
-.0327869	-.03
.52459	.52
-1.60656	-1.61
3.27896	3.28
-5.54098	-5.54
8.39344	8.39
-11.8361	-11.84
15.8689	15.87

LISTING I

```
100 REM ::: HOW BASIC PRINTS NUMBERS :::
110 K= -1
120 FOR I=1 TO 24 STEP 3
```

```
130 N=K*I*I/30.5
140 PRINT N
150 K= -K
160 NEXT I
```

LISTING II

```
100 REM ::: PRINTS A NEAT TABLE :::
110 K= -1
120 D= 2
130 FOR I=1 TO 24 STEP 3
140 N=K*I*I/30.5
150 GOSUB 10000
160 PRINT TAB(20-LEN(NN$)) NN$
170 K= -K
180 NEXT I
190 END
10000 REM ::: SUBROUTINE TO FORMAT D DECIMAL PLACES :::
10010 I$=""
10020 N$= STR$(INT(10^D*N+0.5))
10030 SN$= LEFT$(N$,1)
10040 NN$= RIGHT$(N$,LEN(NN$)-1)
10050 IF LEN(NN$)<D THEN NN$="0"+ NN$: GOTO 10050
10060 D$= RIGHT$(NN$,D)
10070 IF LEN(NN$)>D THEN I$= LEFT$(NN$,LEN(NN$)-D)
10080 NN$= SN$+I$+"."+D$
10090 RETURN
```

=

KEYBOARD READING IN BASIC

By Alan Matzger
960 Guerrero St.
San Francisco,
California 94110

Several months ago, **COLORCUE** told us how we could read our keyboards. Such "reading" is often useful. Indeed, for someone who wants to avoid as many keystrokes as possible, it becomes virtually essential. If I can avoid INPUT statements, my computer and I get along so much more congenially.

The published routine allowed specified time intervals for keyboard entry (and went on to something else if too much time passed); it also permitted keyed in characters NOT to be echoed to the screen. These features certainly do have their advantages, but for most applications I don't need them. The routine I wrote for myself demands echoing and will wait till the next power failure for an entry.

Its several advantages include being written entirely in BASIC; allowing "forbidden" characters, such as commas, quotation marks and control characters, to be accepted without going bananas; and exiting for further processing without having to hit RETURN. Here it is followed by one application.

```
1000 REM       SUBROUTINE TO READ KEYBOARD
1010 F$= "":J=0:REM    BUFFER SET TO NULL STRING AND
1011 REM               CHARACTER POINTER SET TO ZERO
1020 POKE 33278,0:REM  THE LOCATION FOR ANY AND ALL INPUTTED
1021 REM               CHARACTERS -- I NEVER USE CNTL-NULL
1030 F= PEEK(33278):IF F=0 GOTO 1030
1031 REM               WAIT FOR A KEY TO HAVE BEEN STRUCK
1040 IF F=26 AND J>0 THEN PLOT 32,26:J=J-1:F$=LEFT$(F$,J):
```

```
    GOTO 1020:REM      BACKSPACE DELETES LAST CHARACTER BOTH
1041 REM               FROM SCREEN AND BUFFER -- BE CAREFUL
1042 REM               NOT TO BACKSPACE AT LINE'S BEGINNING.
1050 IF F=13 THEN RETURN
1051 REM               THE RETURN KEY WORKS AS USUAL IF NEEDED
1060 J=J+1:F$=F$+CHR$(F)
1061 REM               COUNTER BUMPED ONE AND CHARACTER ADDED
1070 IF J=L THEN RETURN
1071 REM               AUTOMATIC RETURN IF PRESET NUMBER OF
1072 REM               CHARACTERS INPUTTED INTO RECEIVING
1073 REM               VARIABLE.
1080 GOTO 1020:REM     IF NOT, GO BACK FOR ANOTHER CHARACTER.
```

BASIC routines work much slower than those in machine language. However, I type upwards of 70 words a minute and this routine easily keeps up with my blazing speed.

By setting L in the calling program you exit from the subroutine without having to hit RETURN. Such data as dates, numeric choices from a "Menu" and part numbers do have fixed lengths. Here is a "Yes-No" which may find use in some of your programs:

```
1100 L=1:YES= -1:NOE= 0:GOSUB 1000
1101 REM               ONLY ONE CHARACTER WANTED AND THE
1102 REM               VARIABLES ARE PRESET TO TRUE AND FALSE
1110 IF F$="Y" THEN RETURN
1111 REM               "YES" IS ALREADY TRUE. EXIT.
1120 IF F$="N" THEN YES = 0:NOE= -1:RETURN
1121 REM               REVERSE THE "TRUENESSES" AND EXIT.
1130 PLOT 26,32,26:GOTO 1100
1131 REM               UNACCEPTABLE REPLY: ERASE AND TRY AGAIN
```

The following fanciful calling program might show how useful these routines are.

```
5000 PRINT "DO YOU WANT HOT DOGS FOR LUNCH?";
5010 GOSUB 1100
5020 IF NOE THEN FIX HAMBURGERS
5030 REM THE HOT DOG FIXING ROUTINE FOLLOWS
```

NOTE: "NOE" is used because the interpreter will construe "IF NO THEN..." as "IF NOT HEN..." when you have no variable named "HEN" and things won't work right.

=

user group hotline

Compucolor has its followers in several countries. Alberto Nogueras from the Netherlands describes their user group:

In Holland exists the HCC Hobby Computer Club, with more than 3000 members. Within this club are User's Groups for specific types of computers. The "HCC Dutch Intecolor and Compucolor User's Group" is one of them. At this moment, the Compucolor User's Group has 30 members. The members have high

professional backgrounds: 7 software programmers, 4 automation experts, 4 who use the Compucolor at school or on the job, and 11 use the unit for financial and administrative purposes. Only 4 members use the Compucolor as a hobby or to learn about micro-computer hardware, software, and programming and I am one of them. I am an Electronics Engineer and work all over Europe installing and servicing equipment for my company.

The User's Group has a meeting every even month at Utrecht. There, we exchange software and hardware ideas, do some demonstrations, and so on. Our user bulletin appears every two months and we can send in articles to the monthly bulletin of the HCC.

As the users live all over the country, one of the priorities of our group is to communicate via modems and the telephone. For this, the chosen standard baud rate is 300 Baud for all users. Two programs have been written for this purpose: Parity subroutine and Sender (Provisional Version), these programs work very successfully already.

Other software is a Monitor, Real Time Clock (for 50 Hz. with hardware changements), Handshake RS232 bus, Floppy (this program reads all tracks on a floppy and tests the speed, when the FCS gives too much retries, the track is rewritten), Flcoppy (program to copy a disk with only one driver), Lister (lists programs to a printer for BASIC sourcer directly from disk), Mastermind (with 6 different colours which are given by pressing only the color keys), and from myself "Dutch Income Tax 80", and others. Almost all the software comes with Dutch instructions.

=

The Dutch Compucolor User's Group is particularly interested in communicating with users in other countries. To contact the group, write:

Alberto Nogueras
Evertsenstraat 20
5224 HP Den Bosch
The Netherlands

=

keeping it simple

NEW BOOK FOR COMPUCOLOR USERS A WELCOME ADDITION!

Joseph Charles, a Compucolor owner from Hilton, New York, has produced a wonderful book on programming specifically for the Compucolor computer. The book is entitled **BASIC TRAINING FOR COMPUCOLOR COMPUTERS** and provides a wealth of clear instructions, examples, and exercises for those of us who are just learning to use our Compucolors. Mr. Charles did not intend to replace either the **Compucolor Programming Manual** or the **Instruction Manual** with **BASIC TRAINING**, but I'm sure you will find his book a useful adjunct. His 200 page manual is divided into 97 easy-to-digest sections with Index.

By way of a teaser, I have included Mr. Charles' explanation of ASCII Codes so that you may sample some of his

style for yourself. An Order Form for the book has been included in this issue; please order directly from Mr. Charles.

=

ASCII CODES

By Joseph Charles
Dept. B, Box 750
Hilton, New York
14468

ASCII (pronounced "askey") stands for American Standard Code for Information Interchange. Each character that can be used on a computer is represented in the computer by a number. The ASCII code is the standard numerical representation for characters. There are 128 ASCII code values going from 0 to 127. The most pertinent ones for beginning users are those going from 32 to 96. These represent printable alphanumeric and special characters.

There are a number of ways to note the correspondence between the ASCII codes and the characters. The following program will illustrate this correspondence. Clear memory then type this program in:

```
5 REM  THIS IS ASCII1 (ASCII ONE)
8 PLOT 6,3,12:REM  SET FG COLOR,ERASE PAGE
9 PRINT "THESE CHARACTERS CORRESPOND TO ASCII CODES 32 TO 127"
12 PRINT
14 PLOT 6,5
16 FOR I= 32TO 127:PLOT I:NEXT I
18 PLOT 6,3
20 PRINT :PRINT :PRINT
30 PLOT 30:REM  TURN FLAG BIT ON
39 PRINT "WHEN FLAG BIT IS ON, THE COMPUCOLOR II SUBTRACTS"
40 PRINT "96 FROM THE CODE.  THUS CODES 96 TO 127 BECOME"
41 PRINT "0 TO 32. THEREFORE THESE CHARACTERS CORRESPOND TO"
42 PRINT "ASCII CODES 0 TO 32"
43 PRINT
44 PLOT 6,5
45 FOR I= 96TO 127:PLOT I:NEXT I
50 PLOT 29:REM  TURN FLAG BIT OFF
60 PLOT 6,2:PRINT :PRINT :PRINT
```

The ASCII program provides the ASCII code relationships with a minimum of programming complexity. However, it's not that obvious what the ASCII code for each character is. The following program, ASCII2, is a little more complicated, but it provides a nice table which clearly shows the correspondence. Try to understand the program as well as the relationship between the ASCII code and the characters.

```
5 REM  THIS IS ASCII2 (ASCII TWO)
10 PLOT 6,5,12:REM  SET FG COLOR, ERASE PAGE
12 PLOT 15:REM  SET SMALL CHARACTERS
15 PRINT TAB( 6);"CHARACTERS CORRESPONDING TO ASCII
CODES":PRINT
20 FOR I= 0TO 9:PRINT TAB( 5* I+ 5);I;:NEXT I:PRINT
40 FOR K= 0TO 120STEP 10
45 PLOT 6,5
50 PRINT K;
55 PLOT 6,3
```

```
60 FOR I= 0TO 9
65 M= I+ K
70 IF M> 31THEN 150
80 PLOT 30:M= M+ 96:GOTO 160
150 PLOT 29
155 IF M> = 128THEN 190
160 PRINT TAB( 4* I+ 6);:PLOT M
170 NEXT I
175 PRINT
180 NEXT K
190 PLOT 14,6,2:REM  LARGE CHARACTERS, GREEN FOREGROUND
195 PRINT :PRINT :PRINT
200 END
```

Type and run ASCII2 and observe the output on your screen. Notice, for example, that the alphabet has ASCII codes 65 through 90, the decimal digits have codes 48 through 57, etc. There are two sets of 32 special plot characters having ASCII codes 0 through 31 and 96 through 127. The second set, with codes 96 through 127, are obtained directly. The first set, however, having codes 0 to 32, are obtained by setting the flag bit "on" (don't worry about what that means for now) by executing a PLOT 30 instruction and using ASCII codes 96 through 127. When these codes are executed with the flag bit "on", the BASIC interpreter automatically subtracts 96 from the code, thereby converting them to codes 0 through 31. These special plot characters are not standard ASCII characters but are special for the Compucolor II.

=

TOKEN LISTING PROGRAM

By Dennis Martin
P.O. Box 4037
Omaha, Nebraska
68103

In my last article to **COLORCUE** (see "How To POKE Without Getting Jabbed", Dec./Jan. 1980, v.3 #1), I made a brief mention of how the CCII uses tokens to represent command words. This article will attempt to explain what a token is, and includes a program that will generate a complete list of all of the tokens used by the CCII.

A token is what BASIC uses to interpret a command or action to be taken. By using a token instead of the command word itself, the computer makes more efficient use of memory. All command words take only one byte of memory. This is much better than taking seven bytes to store a RESTORE command.

All commands are reserved words and cannot be used as variable names. Although it is advantageous to use descriptive names for variables, this can present problems. As an example, in a checkbook program you might use variables such as: AMOUNT, DEPOSIT, NUMBER, DATE, TO, FOR, etc. TO and FOR are exact command words, and will cause syntax errors. DEPOSIT is not a command word, but it does contain the reserve word POS. This problem can be easily avoided by using short variable names such as: A,B,C,D, etc. This makes the program harder to debug and requires the use of REMark statements to explain what the program is doing. You will find, however, that a descriptive variable name takes up less space than a REMark statement, and words like TO and FOR can be used by changing the "O" to a "0". You must also keep in mind that the Compucolor only recognizes

the first two characters as being the variable name. Thus PAGE and PAYMENT would be treated as the same variable.

The following program will print a complete list of the CCII tokens. This program works by looping through the values for the tokens and then POKEing each value into a REMark statement at the end of the program. A list of tokens is generated by executing a LIST 100 after each POKE.

You might wonder how this program can execute a LIST 100 without listing the rest of the program. When Token is executed, the data table is used to create an entirely new program. Be sure to save this program on disk before typing RUN! List the program after it's done and you will see that there is no similarity between the first and second programs.

```
TOKEN
10 REM  WRITTEN BY                DENNIS L. MARTIN
20 REM                            P.O. BOX 4037
30 REM                            OMAHA, NEBRASKA 68104
40 REM                            (402) 453-6826
50 REM
60 REM  ************************************************
65 REM
70 REM
80 REM          THIS IS A PROGRAM THAT GENERATES THE
90 REM          TOKENS USED BY THE COMPUCOLOR II.  IT
100 REM         ALSO DEMONSTRATES HOW ONE PROGRAM IS
110 REM         USED TO WRITE ANOTHER PROGRAM INTO
120 REM         MEMORY.  BE SURE TO SAVE THIS ON DISK
130 REM         BEFORE TYPING 'RUN'!!  WHEN THIS IS
140 REM         RUN, YOU WILL NO LONGER HAVE IT IN
150 REM         MEMORY!!  YOU MIGHT WANT TO LIST THE
160 REM         NEW PROGRAM TO SEE EXACTLY HOW IT
170 REM         DISPLAYS THE TOKENS.
180 REM
190 REM ************************************************
1000 Z= 33433
1005 DIM D(250)
1010 RESTORE 2000
1020 FOR Q= 1TO 225:READ D(Q):NEXT
1025 FOR T= 1TO 225
1030 POKE Z,D(T)
1040 Z= Z+ 1
1050 IF Z= 33640THEN 1500
1060 NEXT
1070 END
1500 RUN
1520 END
2000 DATA 0,166,130,10,0,88,172,51,51,54,50,53,0
2100 DATA 175,130,11,0,65,172,53,52,0
2200 DATA 185,130,15,0,66,172,49,50,55,0
2300 DATA 195,130,20,0,65,172,65,164,49,0
2400 DATA 204,130,30,0,149,88,44,66,0
2500 DATA 233,130,35,0,138,65,172,53,56,161,65,172,52
2600 DATA 56,58,149,88,165,53,44,189,40,88,165,53,41
2700 DATA 164,49,0,15,131,36,0,138,189,40,88,165,53
```

```
2800 DATA 41,171,172,53,56,161,149,88,165,53,44,52,56
2900 DATA 58,149,88,165,54,44,189,40,88,165,54,41
3000 DATA 164,49,0,26,131,40,0,149,88,165,52,44,65
3100 DATA 0,35,131,50,0,151,49,48,48,0,45,131
3200 DATA 60,0,66,172,66,164,49,0,59,131,65,0,138
3300 DATA 66,171,49,57,55,161,56,48,0,67,131,70
3400 DATA 0,136,50,48,0,77,131,80,0,146,49,53
3500 DATA 58,128,0,91,131,100,0,142,32,49,50,55
3600 DATA 32,61,32,64,0,0,0,0
3700 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0
```

=

EDITOR'S NOTE

To recap: the Compucolor reduces BASIC commands to one byte symbolic representations, called tokens, in order to reduce space and make programs run more efficiently. Here's a list of Compucolor BASIC tokens.

COMPUCOLOR II BASIC TOKENS

By R.M. Manazir
13 Grandview St.
Southwick, Mass.
01077

HEX VALUE	DECIMAL VALUE	COMMAND	KEYBOARD KEY & COMMAND KEY
80	128	END	@
81	129	FOR	A
82	130	NEXT	B
83	131	DATA	C
84	132	INPUT	D
85	133	DIM	E
86	134	READ	F
87	135	FILE	G
88	136	GOTO	H
89	137	RUN	I
8A	138	IF	J
8B	139	RESTORE	K
8C	140	GOSUB	L
8D	141	RETURN	M
8E	142	REM	N
8F	143	GET	O
90	144	OUT	P
91	145	PUT	Q
92	146	PLOT	R
93	147	SAVE	S
94	148	LOAD	T
95	149	POKE	U
96	150	PRINT	V
97	151	LIST	W
98	152	CONT	X
99	153	CLEAR	Y
9A	154	DEF	Z
9B	155	WAIT	[
9C	156	ON	\
9D	157	TAB(	]
9E	158	TO	^
9F	159	FN	_
A0	160	SPC(	0
A1	161	THEN	1
A2	162	NOT	2
A3	163	STEP	3

A4	164	+	4
A5	165	-	5
A6	166	*	6
A7	167	/	7
A8	168	^	8
A9	169	AND	9
AA	170	OR	:
AB	171	>	;
			& CONTROL KEY
AC	172	=	,
AD	173	<	-
AE	174	SGN	.
AF	175	INT	/
B0	176	ABS	0
B1	177	CALL	1
B2	178	FRE	2
B3	179	INP	3
B4	180	POS	4
B5	181	SQR	5
B6	182	RND	6
B7	183	LOG	7
B8	184	EXP	8
B9	185	COS	9
BA	186	SIN	:
BB	187	TAN	;
			& COMMAND KEY
BC	188	ATN	,
BD	189	PEEK	-
BE	190	LEN	.
BF	191	STR$	/
			& CONTROL KEY
C0	192	VAL	F0
C1	193	ASC	F1
C2	194	CHR$	F2
C3	195	LEFT$	F3
C4	196	RIGHT$	F4
C5	197	MID$	F5

=

advanced applications

UTILITY ROUTINES

While programming, one often finds the need for functions and routines that are not provided either in the programming language or in the hardware. The programmer finds that hardware limitations are especially noticeable at the assembly language programming level. The 8080 instruction set does not include such instructions as a double-register compare or a double-register subtract. We shall discuss these and several other utility routines.

The routines that will be presented here will, for the most part, be uncommented. Those without comments are straight forward and their exact explanations will be left to the reader. It should also be noted that these routines do appear in the system software and the labels used are the actual labels. The addresses for these routines in both versions of system software will appear at the end of this section.

The first group of routines will just be described as to their function.

CRLF - lists a carriage return (CR) and a linefeed (LF) to the output device using the routine 'LO'.

LBYT - takes the byte value in the A register and lists it as two hexadecimal characters.

WATL - this is a 'wait' routine of 20 milliseconds per count with the count being in register A.

WATS - this is a 'wait' routine of 0.5 milliseconds per count with the count being in register A.

The next group consists of routines that can have several configurations, not all of which are provided in the system software.

The routine 'CMPHD' is a double-register compare. The value in H&L is compared to the value in D&E and the following conditions are returned.

H&L less than D&E - carry flag is set <C>
H&L equal to D&E - zero flag set <Z>
H&L greater than D&E - neither flag set

The advantage to making a routine callable can be seen in 'CMPHD' because it has more than one possible exit. When this is true, a callable routine is easier to implement and requires less code than if it were programmed in-line.

```
CMPHD:  ; Compare for HL <=> DE  (C,Z,0)
        MOV     A,H
        CMP     D
        RNZ
        MOV     A,L
        CMP     E
        RET
```

The routine 'CMPDH' is handled the same way except that H&L and D&E are switched. This can also be extended to routines such as 'CMPBH', 'CMPDB', etc.

The routine 'MOVDH' moves a number of bytes from the location pointed to by H&L to the location pointed to by D&E. The count of the bytes to be moved is passed to the routine in the B register.

```
MOVDH:  ; Move B bytes from (HL) to (DE)
        MOV     A,M
        STAX    D
        INX     H
        INX     D
        DCR     B
        JNZ     MOVDH
        RET
```

The routine 'MOVHD' is similar except that D&E point to the source and H&L point to the destination.

The routine 'SUBHD' subtracts the value in D&E from the value in H&L and places the result in H&L.

```
SUBHD:  ; HL <== HL - DE
        MOV     A,L
        SUB     E
        MOV     L,A
        MOV     A,H
        SBB     D
        MOV     H,A
        RET
```

The last routine, 'B2HEX', takes the lower four bits in the A register and converts it to an ASCII hexadecimal digit. This routine is used by 'LBYT'.

```
B2HEX:  ; Convert low nibble in A to Hex char.
        ANI     0FH     ; CLEANSE VALUE
        ADI     90H
        DAA
        ACI     40H
        DAA
        RET             ; A = ASCII HEX DIGIT
```

The following is a list of the routines discussed and their locations in the system software.

	V6.78	V8.79
CRLF	338BH	17C1H
LBYT	339BH	17D1H
B2HEX	33AAH	17E0H
WATS	341CH	1852H
WATL	3429H	185FH
MOVDH	343BH	1871H
MOVHD	3444H	187AH

CMPHD	344DH	1883H
CMPDH	3453H	1889H
SUBHD	3459H	188FH

=

nuts & bolts

Part of the reason why computers are so attractive is because the number of applications possible is limited only by your own imagination and inventiveness. You can use your Compucolor to distinguish between, and control, levels of sound, light, heat, cold, air, voltage, amperage, and a multitude of other external conditions.

EDITOR'S NOTE

Announcing our new hardware and interface section -- at last! Many of you have been asking for a hardware column to answer questions and provide hardware ideas. Here's your chance to ask all the hardware questions that you've been saving and also to share your hard-won knowledge with others. I hope that you will feel free (obligated!) to send me your special hardware projects so that we can share the wealth -- that "insignificant" little hardware or conversion program that you've developed may be just what someone else needs right now! One thing that we have had many requests for are conversions -- to anything!

We need your input. Please remember to document projects and articles fully (pictures and diagrams too, if you think they will help demonstrate your point). Send your questions and projects to me at my editorial office. Our first Nuts & Bolts article is on making custom character sets.

=

CUSTOM CHARACTER SETS

By Trevor Taylor
200A Foxridge Apts.
Blacksburg, Va.
24060

I use my CCII mainly as a communications terminal on IBM and Honeywell mainframes. The IBM machine prints error messages in lower-case and after trying to decipher various chess pieces and miscellaneous blobs, I decided it was time to get myself some lower-case characters.

What I did was to make my own lower-case character generation ROM and replace UF6 on the Logic Board. (It is in a socket, so this is easy.) Of course, it is possible to make any character set you desire -- Greek if you like. All you need is a 2708 EPROM (UV-Erasable Programmable Read Only Memory), and I bought one for $12.95 at my local electronics store (they sell Jim-Pak components from Jameco Electronics in California). There is one catch however, you MUST have access to a 2708 EPROM Programmer. (Warning: Static electricity kills EPROMs! Always touch the metal case of the disk drive in your CCII before handling a ROM.)

Doing it this way does not allow you to have both the special characters and lower case at the same time. The lower case option from Compucolor does, but it costs more. You can still play games because of the way Compucolor designed their character set. With lower case characters instead of special characters, the chess pieces correspond as follows:

a = knight s = bishop k = king p = pawn

q = queen r = rook (castle)

and for the playing cards:

c = clubs d = diamonds h = hearts s = spades

So, how do you make your own character set? On the Logic Board you will find two ROMs, labelled UF6 and UF7, at the top left-hand side. UF7 contains all the "characters" for graphics, so you should leave this alone. UF6 has all the alphanumerics.

This is a 1024x8 bit ROM (1K bytes) with all the data for 128 characters, i.e. 8 bytes for each character. It works as follows: Take the ASCII code for the character and multiply it by 8, then read 8 bytes from the ROM starting at this address. For example, the code for "y" is 121 (in decimal), so 8x121 = 968. Bytes 968 through 975 are the dot pattern for 'y'. (in hex, this is bytes 3C8-3CF.) For my ROM, the data is as follows:

Address	Contents= Hex	Binary
3C8	00	00000000
3C9	00	00000000
3CA	44	01000100
3CB	44	01000100
3CC	44	01000100
3CD	3C	00111100
3CE	04	00000100
3CF	38	00111000

Does the binary representation look like anything? The CCII uses an 8x6 matrix for characters, i.e. 8 rows by 6 columns. If we throw away the two low order bits of each byte (they are zeros anyway) and put dots on the screen everywhere there is a one, we get:

row	column 1	2	3	4	5	6
1						
2						
3		*				*
4		*				*
5		*				*
6			*	*	*	
7						*
8			*	*	*	

However, the CRT Controller does not do it this way. It fills in all the dots for row one of every character on a line, scanning from left to right across the whole screen, then does the same for row two, etc.

For most characters, row 8 is empty, which results in the spaces between lines on the screen. If there is a capital 'O' on the next line, on the screen immediately below the 'y', the tail of the 'y' appears to merge with the top of the 'O'. This does not really make it more difficult to read.

Almost without exception, the alphanumerics have nothing in column 1. This gives space between letters on a line. Without it, they would all blur together.

Anybody who can work with binary and hex can design a character set, but if you can't be bothered, I will send you copies of the data sheets for the lower-case characters. Please include $1 to cover photocopying and postage (and don't forget to include a return address!).

One last thing, BASIC and FCS do not recognize lower case. However, you can put lower case characters inside quotes in PRINT statements and in strings. I sometimes answer 'y' to a yes or no question, which is not the same as 'Y' and is taken to mean NO! The CAPS LOCK key solves this problem.

=

The EROM 2708 must be a fast type 350 ns or faster. You take a chance of fading characters if you use a slower part (normal parts are 450 ns).

EDITOR'S NOTE

=

input

BIORHY ENHANCEMENTS

Bill Greene
RT3 Box 00-200
Byron, Ga. 31008

Following are seven improvements that I have made to BIORHY, the biorhythm program on the Sampler Sof-Disk. They are arranged in order of simplest to most complex and may be added independently. Nos. 6&7 are in combined form and assume that 1 thru 5 have been implemented.

1. To print the starting day in the title heading. (included in 5)

```
170 PLOT 3,16,0: D$=STR$(DC)
```

2. To correct colors of Mental and Physical Cycles.

```
20 C(0)=6:C(1)=2:C(2)=1
```

3. To prevent BS ERROR when start date >Dec 3 and <Jan 1.

```
3028 IF MC=13 THEN MC=1:N=1:Y1=Y1+1
3029 IF DC<MX(MC) THEN DC=DC+1
3027 IF DC= MX(MC)THEN DC= 0:MC= MC+ 1:N= N+ 1
```

4. To continue current chart for another month or to start a new chart without returning to MENU.

```
3080 IF LEFT$(YN$,1)="Y"THEN SD=ED+1:ED=SD+28:GOTO165
3090 PLOT 28,11:INPUT"ANOTHER CHART ?";YN$
3095 IF LEFT$(YN$,1)="Y"THEN MX(2)=28:GOTO90
4000 LOAD"MENU";1":RUN5
```

If 5. is to be implemented, use the following line 3080:

```
3080 IF LEFT$(YN$,1)="Y"THEN SD=ED+1:ED=SD+30:GOTO165
```

5. To display 31 days instead of 29 days.

```
38 YS=0:1Y=4
160 ED=SD+30
161 YY=YS+1Y*30
165 PLOT12,6,4,2,250,S,LL-1,YY,LM,YY,LH+1,YY,255,3,16,0
166 FOR I=1TO31:CD(1)=W:W=W+1
168 IF W=8THENW=1
169 NEXT I
170 D$=STR$(DC)
3005 FOR I=1TO31
3080 (SEE NO.4)
```

6. The following lines will calculate the number of days between any two calendar dates after the year 1582, and the day of the week after the year 1582, and

7. Corrects blinking of critical days. If a cycle crosses the center line midway between two dates, both dates will blink, otherwise only the nearest date will blink.

Delete line 143
Delete lines 1830 to 2160 inclusive
Add the following lines:

```
185 PLOT 3,64,0:IF N= 2THEN GOSUB 5000
1830 IF M< 3THEN N1= 0:N2= INT ((Y1- 1)/ 4):GOTO 1870
1850 N1- INT (.4* M+ 2.3)
1860 N2= INT (Y1/ 4)
1870 N3= INT (.75* (INT (Y1/ 100)+ 1))
1880 J2= 365* Y1+ (31* (M- 1))+ D+ N2- N1- N3
1900 W= J2- (INT (J2/ 7)* 7)
1910 IF W= 0THEN W= 7
1920 RETURN
```

Delete line 2301
Delete lines 2311 to 2440 inclusive.

Add the following lines:

```
2312 IF XX- LM> 5.5THEN 2320
2314 IF LM- XX> 5.5THEN 2320
2315 IF CD(1)> 64THEN 2320
2316 CD(1)= CD(1)+ 64
2320 PLOT 253:PLOT YS:PLOT XX:PLOT 242
2410 FOR X= SD+ 1TO ED
2430 D9= 2* (X- BD)* 3.1415927
2435 XX= INT (SIN (D9/ CY)* LR+ LM)
2437 IF XX- LM> 5.5THEN 2445
2438 IF LM- XX> 5.5THEN 2445
2439 IF CD(X- SD+ 1)> 64THEN 2445
2440 CD(X- SD+ 1)= CD(X- SD+ 1)+ 64
```

Delete line 3020
Delete line 3026
Delete lines 5000 and 5001
Add the following lines:

```
5000 IF Y1/ 400- INT (Y1/ 400)= 0THEN 5030
5010 IF Y1/ 100- INT (Y1/ 100)= 0THEN 5040
5020 IF Y1/ 4- INT (Y1/ 4)<> 0THEN 5040
5030 MX(2)= MX(2)+ 1:RETURN
5040 RETURN
```

=

KALEIDOSCOPE

This program runs on a Compucolor II to produce a different design every time it runs. To get a changing, kaleidoscopic design, "LENGTH OF RUN" should be set to 50 or more.

By George Herman
724 Ash Street
Bowling Green,
Ohio 43402

****

```
10 PLOT 12: PRINT: PRINT: PRINT
20 PRINT TAB (26); "KALEIDOSCOPE"
30 REM          WRITTEN BY GEORGE HERMAN
40 REM          BOWLING GREEN, OHIO, JUL 1, 1979
50 PRINT: PRINT: PRINT
60 INPUT "LENGTH OF RUN?"; N
70 PLOT 12,6,1,2,10,10
80 PLOT 242,10,118,118,118,118,10,10,10,255
90 FOR I= 1 TO N
100 FOR J= 1 TO 7
110 PLOT 6,J:  REM     CHANGES COLOR OF EACH DOT
120 X=RND(1)*50
130 Y=RND(1)*X
140 PLOT 2,X+14,Y+14,114-X,114-Y,255
150 PLOT 2,Y+14,X+14,114-Y,114-X,255
160 PLOT 2,X+14,114-Y,114-X,Y+14,255
170 PLOT 2,114-Y,X+14,Y+14,114-X,255
180 NEXT J
190 NEXT I
200 END
```

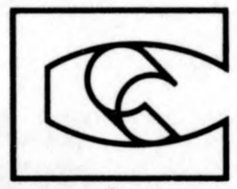
=

**********ERRATA**********

```
;***************************************************************
;
;       PRINTER DRIVER FOR SIMPLE MINDED PRINTERS
;       USING CLEAR-TO-SEND
;
;       EXPANDS TABS
;       NOTICES #CHARS ON LINE
;       AND #LINES ON PAGE
;       PRINTS HEADER MSG

PRINT:  CALL    SAVE            ; SAVE ALL
        CPI     FF              ; FORM FEED?
        JZ      FFEED           ; DO IT
        CPI     TAB             ; SEE IF TAB
        JZ      TABOUT          ; SEND BLANKS FOR DUMB PRINTERS
        CPI     LF              ; LF? END OF LINE
        JNZ     COUNT           ; NO PRINT IT
        CALL    SEND            ; PRINT IT ANYWAY
        LXI     H,CCOUNT        ; RESET CHAR COUNT
        MVI     M,0             ; TO ZERO
        LXI     H,LCOUNT        ; GET LINE COUNT
        INR     M               ; BUMP IT
        LDA     WHITE           ; GET MARGIN SIZE
        MOV     B,A             ; SAVE IT
        LDA     PSIZ            ; GET PAGE SIZE
        SUB     B               ; SUB MARGIN
        CMP     M               ; THERE YET?
        RNZ                     ; NO THATS ALL TO DO
        CALL    MARGIN          ; YES - SEND SOME BLANK LINES
        MVI     M,0             ; NOW AT TOP OF PAGE
TOF:    LXI     H,HMSG          ; SO... HEADER MESSAGE
HEADER:MOV      A,M             ; GET CHAR
        ORA     A               ; END OF HEADER?
        INX     H               ; TO NEXT CHAR
        CNZ     PRINT           ; SEND
        JNZ     HEADER          ; REPEAT
MARGIN:LDA      WHITE           ; AND SOME MORE BLANK LINES
MRG01:  PUSH    PSW             ; SAVE COUNT
        MVI     A,LF            ; KEEP THOSE LINE FEEDS
        CALL    PRINT           ; GOING OUT
        POP     PSW             ; RECOVER COUNT
        DCR     A               ; TILL WE HAVE ENOUGH MARGIN
        JNZ     MRG01           ; KEEP IT UP
        RET                     ; WHEW! ALL THIS FROM ONE CHAR!

FFEED:  MVI     A,LF            ; FORM FEED  BY LINE FEEDS
        CALL    SEND            ; SEND IT
        LXI     H,LCOUNT        ; COUNT IT
        INR     M               ; ONE MORE LINE
        LDA     PSIZ            ; SEE IF END OF PAGE
        XRA     M               ; TRICKY ZERO <A> IF SAME
        JNZ     FFEED           ; NO
```

```
        MOV     M,A             ; YES, USE TRICKY ZERO
        JMP     TOF             ; AND PRINT HEADER,BLANK LINES

COUNT:  CALL    SEND            ; SEND IT
        LXI     H,CCOUNT        ; ONE MORE CHAR ON THIS LINE
        INR     M               ;
        LDA     LSIZ            ; MAX LINE SIZE
        CMP     M               ; OUR COUNT
        RNZ                     ; FORCE LF
        MVI     A,CR            ; SEND CR
        CALL    PRINT           ; SEND LF,RESET COUNT
        MVI     A,LF            ; TEST FOR END OF PAGE, ECT
        JMP     PRINT           ; AND RETURN

SEND:   PUSH    PSW             ; SAVE CHAR
        MOV     E,A             ; E FOR SERVICE ROUTINE
        CALL    SlOUT           ; TEST CLEAR TO SEND THEN SEND
        POP     PSW             ; CHAR BACK
        CALL    CO              ; DISPLAY
        RET

TABOUT: MVI     A,' '           ; SEND SPACES
        CALL    PRINT           ;
        LDA     CCOUNT          ; TILL CCOUNT=
        ANI     7               ; EVEN MULTIPLE OF 8
        RZ                      ;
        JMP     TABOUT          ;

;       DATA AREA PRINTER DRIVER
;
CCOUNT:         DB      0               ; CURRENT CHAR COUNT
LCOUNT:         DB      0               ; CURRENT LINE COUNT

;       RUN TIME PRAMS - DEFAULT VALUES
;
LSIZ:   DB      80              ; MAX CHAR ON LINE
PSIZ:   DB      66              ; MAX LINES ON PAGE
WHITE:  DB      3               ; NUMBER OF BLANK LINES AT TOP OF
PAGE

HMSG:   DB      ' COMPUCOLOR II PRINT',TAB,TAB,' V2.0'
        DB      CR,LF,' ---------- -- -----',TAB,TAB,'
----',CR,LF,0

;       END SIMPLE MINDED PRINTER DRIVER
;
;*****************************************************************
```

=



Compucolor® Corporation
Intecolor Drive
225 Technology Park/Atlanta
Norcross, Georgia 30092
Telephone 404/449-5996